A Load Balancing Problem

- We are given n items to be placed in m bins
  - Item i has positive integer weight w_i, $1 \leq i \leq n$
  - We wish to assign each item to a bin in a way that minimizes the maximum bin load
  - Remark: The "load" of a bin is the total weight of the items assigned to it
- We will prove that the decision version of this problem, call it BALANCE, is NP-complete
  - Given a threshold B, is there a way to assign the items to the bins so that every bin has load at most B?
  - It is easy to argue that BALANCE belongs to NP

The Subset Sum Problem

- In the SUBSET SUM problem, we are given n integers $x_1, \ldots, x_n$ and an integer target T, and we wish to determine whether there is a subset I of $\{1, \ldots, n\}$ such that $\sum_{i \in I} x_i = T$
- It is easy to argue that SUBSET SUM belongs to NP
- The text uses a reduction from 3-SAT to prove that SUBSET SUM is NP-complete

The Partition Problem

- In the PARTITION problem, we are given n positive integers $x_1, \ldots, x_n$, and we wish to determine whether there is a subset I of $\{1, \ldots, n\}$ such that

$$\sum_{i \in I} x_i = \sum_{i \in \{1,\ldots,n\} \setminus I} x_i = \frac{1}{2} \sum_{1 \le i \le n} x_i$$

- It is easy to see that PARTITION belongs to NP
- We will use a reduction from SUBSET SUM to prove that PARTITION is NP-complete

A Reduction from SUBSET SUM to PARTITION

- Let X be an instance of SUBSET SUM with positive integers $x_1, \ldots, x_n$ and target T
  - Let S denote $\sum_{1 \leq i \leq n} x_i$
  - Since target T is effectively the same as target $S - T$, we can assume without loss of generality that $T \geq S/2$
  - If $T = S/2$ then X corresponds directly to an instance of PARTITION (simply drop the target T), so we can assume that $T > S/2$
- Let Y denote the PARTITION instance with $n + 1$ positive integers $x_1, \ldots, x_{n+1}$ where $x_{n+1} = 2T - S$
- We claim that X is a positive instance of SUBSET SUM if and only if Y is a positive instance of PARTITION

"Only If" Direction

- Assume that X is a positive instance of SUBSET SUM
- Hence there is a subset A of $\{1, \ldots, n\}$ such that $\sum_{i \in A} x_i = T$
- Let B denote $\{1, \ldots, n+1\} \setminus A$
- Thus $\sum_{i \in B} x_i = (S - T) + (2T - S) = T$
- Thus Y is a positive instance of PARTITION

"If" Direction

- Assume that Y is a positive instance of PARTITION
- Thus we can partition $\{1, \ldots, n+1\}$ into two sets A and B such that $\sum_{i \in A} x_i = \sum_{i \in B} x_i = [S + (2T - S)]/2 = T$
- Assume without loss of generality that $n+1$ belongs to B
- Thus A is a subset of $\{1, \ldots, n\}$ such that $\sum_{i \in A} x_i = T$
- Thus X is a positive instance of SUBSET SUM

A Reduction from PARTITION to BALANCE

- Let X be an instance of PARTITION with positive integers $x_1, \ldots, x_n$
  - Let S denote $\sum_{1 \leq i \leq n} x_i$
  - We can assume without loss of generality that S is even
- Let Y be the instance of BALANCE with item weights $x_1, \ldots, x_n$, $m = 2$ bins, and threshold B equal to $S/2$
- Observe that X is a positive instance of PARTITION if and only if Y is a positive instance of BALANCE
- Thus BALANCE is NP-complete, even in the special case $m = 2$

The Class of NP-Hard Problems

- We say that a problem X is NP-hard if existence of a polynomial-time algorithm for X implies that $\mathrm{P} = \mathrm{NP}$
  - Importantly, the problem X need not be a decision problem
  - For example, the problem of determining a maximum independent set of a graph is NP-hard
- It is easy to see that the load balancing problem is NP-hard
- It is natural to seek approximation algorithms for such NP-hard optimization problems

Approximation Algorithms for Optimization Problems

- We typically seek approximation algorithms achieving a good approximation ratio
  - For a minimization problem, we say that an algorithm achieves an approximation ratio of α if it is guaranteed to produce a solution with objective function value at most α times optimal
  - For a maximization problem, we say that an algorithm achieves an approximation ratio of α if it is guaranteed to produce a solution with objective function value at least $1/\alpha$ times optimal
  - Thus the approximation ratio is at least 1, and an optimal algorithm achieves an approximation ratio of 1

Approximability of NP-Hard Optimization Problems

- Given an NP-hard optimization problem, we seek to design an approximation algorithm with an approximation ratio as close to 1 as possible
- As we have seen, many NP-complete decision problems have an associated NP-hard optimization version
  - It turns out that these optimization problems vary widely in terms of polynomial-time approximability
  - For example, in some cases we can give a polynomial-time algorithm that achieves an approximation ratio of $1 + \varepsilon$ for any constant $\varepsilon > 0$
  - In other cases we can show that for any $\varepsilon > 0$, no polynomial-time algorithm achieves an approximation ratio of $n^{1-\varepsilon}$ unless $\mathrm{P} = \mathrm{NP}$

Greedy Approximation Algorithms

- For many NP-hard optimization problems, we can use a simple greedy algorithm to achieve a good approximation ratio
- Today we will analyze two greedy algorithms for the load balancing problem
- The first of these two algorithms, which we denote $\mathcal{A}$, processes each successive item (from 1 to n) by placing it in a least-loaded bin
  - We will prove that algorithm $\mathcal{A}$ achieves an approximation ratio of 2
- The second algorithm, denoted $\mathcal{B}$, is the same as $\mathcal{A}$ except that it processes the items in nonincreasing order of weight
  - We will prove that algorithm $\mathcal{B}$ achieves an approximation ratio of $4/3$

Algorithm $\mathcal{A}$: An Upper Bound of 2

- Fix an instance of the load balancing problem, and let L^* denote the minimax bin load
- Suppose that at some step of algorithm $\mathcal{A}$, we add an item i with weight w to a bin j with load L
  - After this step, bin j has load $L + w$
  - Since L is the minimum load before the step, the total weight of the items is at least mL and hence $L \leq L^*$
  - Since item i needs to be placed in some bin, we have $w \leq L^*$
  - Hence $L + w \leq 2L^*$
  - Thus algorithm $\mathcal{A}$ achieves an approximation ratio of 2

Algorithm $\mathcal{A}$: A Lower Bound of $2 - \varepsilon$

- Consider an instance with $n = k(k+1) + 1$ items and $m = k + 1$ bins where $w_i = 1$ for $1 \leq i \leq k(k+1)$ and $w_n = k + 1$
- After algorithm $\mathcal{A}$ has processed the first $k(k+1)$ items, each bin has a load of k
  - Thus, after the last step, the max bin load is $2k + 1$
- It is possible to achieve a max bin load of $k + 1$
  - Use one bin to hold the item of weight $k + 1$
  - Use each of the remaining k bins to hold $k + 1$ unit-weight items
- For any constant $\varepsilon > 0$, we can ensure that the approximation ratio exceeds $2 - \varepsilon$ by choosing k sufficiently large

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{3}{2}$

- Fix an instance of the load balancing problem, and let L^* denote the minimax bin load
- Suppose that at some step of algorithm $\mathcal{B}$, we add an item i with weight w to a bin j with load L
  - After this step, bin j has load $L + w$
  - If $L = 0$, then $L + w = w \leq L^*$
  - Now assume $L > 0$
    - There are at least $m + 1$ jobs with weight at least w
    - Hence $2w \leq L^*$, so $w \leq L^*/2$
    - Since $L \leq L^*$, we conclude that $L + w \leq (3/2)L^*$

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$

- Fix an instance of the load balancing problem, and let L^* denote the minimax bin load
- Assume without loss of generality that $w_1 \geq \ldots \geq w_n$
- We say that item i is heavy if $w_i > L^*/3$, and is light otherwise
- Let h denote the number of heavy items
- Claim 1: No optimal solution puts more than two heavy items into the same bin
  - The load of a bin with three heavy items exceeds L^*
- Claim 2: $h \leq 2m$
  - Immediate from Claim 1

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$ (cont'd)

- Partition the execution of algorithm $\mathcal{B}$ into a "heavy phase", in which we place the heavy items $w_1, \ldots, w_h$, and a "light phase", in which we place the remaining items
- Suppose that at some step of algorithm $\mathcal{B}$ during the light phase, we add an item i with weight w to a bin j with load L
  - Since $L \leq L^*$ and $w \leq L^*/3$, we have $L + w \leq (4/3)L^*$
- It remains to consider the heavy phase
  - Let L^{**} denote the minimax load for the subinstance with m bins and only the h heavy items
  - Thus $L^{**} \leq L^*$

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$ (cont'd)

- In what follows, we complete the proof by establishing the following result
- Lemma 1: At the end of the heavy phase, the max bin load is L^{**}
- By Claim 2, $h \leq 2m$
- If $h \leq m$, it is easy to see that the claim of Lemma 1 holds
- For the remainder of the proof, assume that $h = m + s$ where $1 \leq s \leq m$
- We partition the h heavy items into three sets
  - The set A of items 1 through $m - s$
  - The set B of items $m - s + 1$ through m
  - The set C of items $m + 1$ through $m + s$

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$ (cont'd)

- Let $\mathcal{X}^*$ denote the set of all ways to assign the heavy items to the bins such that the max bin load is L^{**}
  - These are the optimal solutions to the m-bin subinstance corresponding to the heavy items $w_1, \ldots, w_h$
  - We know that no such assignment places more than two items in the same bin
- Let $\mathcal{X}$ denote the set of all ways to assign the heavy items to the bins such that there are at most two items in each bin
- Our plan is to use a series of exchange arguments to prove that the assignment produced by the heavy phase of algorithm $\mathcal{B}$ belongs to $\mathcal{X}^*$

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$ (cont'd)

- We say that an assignment in $\mathcal{X}$ is A-nice if for any item i in A, no other item is assigned to the same bin as item i
- Claim 3: There is an A-nice assignment in $\mathcal{X}^*$
  - Let σ be an assignment in $\mathcal{X}^*$
  - If σ is not A-nice, then some item i in A shares a bin with another item i', while some item i'' in $B \cup C$ has its own bin
  - The max load does not increase if we modify σ by moving item i' to the bin with item i''
  - This modification reduces by 1 the number of items in A that share a bin with another item
  - We can repeat this argument until we reach an A-nice assignment in $\mathcal{X}^*$

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$ (cont'd)

- We say that an assignment in $\mathcal{X}$ is B-nice if no two items in B are assigned to the same bin
- Claim 4: There is an assignment in $\mathcal{X}^*$ that is A-nice and B-nice
  - By Claim 3, there is an A-nice assignment σ in $\mathcal{X}^*$
  - If σ is not B-nice, then there are items i_0 and i_1 in B that share a bin, and items i_2 and i_3 in C that share a bin
  - The max load does not increase if we modify σ by exchanging items i_1 and i_3; moreover, the modified assignment is A-nice
  - We can repeat this argument until we reach an assignment in $\mathcal{X}^*$ that is A-nice and B-nice

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$ (cont'd)

- Observe that if an assignment in $\mathcal{X}$ is A-nice and B-nice, then each of the s items in B shares a bin with one of the s items in C
- We say that an assignment in $\mathcal{X}$ is nice if it is A-nice, B-nice, and for every pair of items i and i' in B such that $w_i > w_{i'}$, we have $w_j \leq w_{j'}$ where item j (resp., j') in C shares a bin with item i (resp., i')
- Claim 4: Any two nice assignments in $\mathcal{X}$ have the same distribution of bin loads
  - Each item in A has its own bin
  - Each of the remaining bins contains one item from B
  - The ith heaviest item in B is paired with the ith lightest item in C (under some tie breaking) for $1 \leq i \leq s$

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$ (cont'd)

- Claim 5: There is a nice assignment in $\mathcal{X}^*$
  - By Claim 3, there is an assignment σ in $\mathcal{X}^*$ that is A-nice and B-nice
  - If σ is not nice, there are items i and i' in B such that $w_i > w_{i'}$ and $w_j > w_{j'}$ where item j (resp., j') in C shares a bin with item i (resp., i')
  - The max load does not increase if we modify σ by exchanging items j and j'; moreover, the modified assignment is A-nice and B-nice
  - We can repeat this argument until we reach a nice assignment in $\mathcal{X}^*$

Algorithm $\mathcal{B}$: An Upper Bound of $\frac{4}{3}$ (cont'd)

- It is easy to see that algorithm $\mathcal{B}$ produces a nice assignment in $\mathcal{X}$ at the end of the heavy phase
- By Claims 4 and 5, we conclude that $\mathcal{B}$ produces an assignment in $\mathcal{X}^*$ at the end of the heavy phase
- This completes the proof of Lemma 1

Algorithm $\mathcal{B}$: A Lower Bound of $\frac{4}{3} - \varepsilon$

- Exercise: As in the $2 - \varepsilon$ lower bound argument for algorithm $\mathcal{A}$, identify a suitable family of "bad" instances for algorithm $\mathcal{B}$